ซอฟต์แวร์
บุญแสง สิริรัตน์ชูวงศ์

SOFT MAIL

การสร้างจอภาพเสมือน

เพิ่มจำนวนบรรทัดให้กับจอภาพ เพื่อประสิทธิภาพในการแสดงผลของข้อมูล

จอภาพในโหมดตัวอักษรขนาด 80×25 ตัวอักษรที่ใช้กันอยู่ ทำให้เกิดข้อจำกัดของขนาดข้อมูลที่จะแสดงให้ผู้ใช้ได้เห็น ทำให้เราต้องคอยคำนึงถึงข้อมูลที่เขียนบนจอ เพราะจะต้องอยู่ในขอบเขตที่จำกัด ดังนั้นถ้าหากมีจอภาพเสมือนที่สามารถจะเขียนหรือแสดงข้อมูลต่าง ๆ ได้ โดยมีขนาดตามที่ต้องการ แล้วจึงนำข้อมูลจากจอภาพเสมือนออกมาแสดงทางจอภาพจริงในภายหลัง จะทำให้การแสดงผลมีประสิทธิภาพยิ่งขึ้น ตัวอย่างในบทความนี้จะแสดงการสร้างจอภาพเสมือนที่มีจำนวนบรรทัดของจอภาพเสมือนตามที่เราต้องการ แต่ยังคงใช้ความกว้างของจอภาพเสมือนขนาด 80 ตัวอักษร อย่างไรก็ตามผู้อ่านสามารถนำไปดัดแปลงตามความประสงค์ของผู้อ่านได้

การใช้ยูนิต Virtual

ยูนิต Virtual ในรูปที่ 1 มีโพรซีเดอร์ในส่วนอินเตอร์เฟสมีอยู่ 4 โพรซีเดอร์เท่านั้น โพรซีเดอร์ทั้ง 4 นี้สามารถใช้งานยูนิต Virtual ได้แล้ว โดยมีหน้าที่ดังนี้

```
1: Unit Virtual;
2: Interface
3: Uses Crt,Dos;
4:
5: Procedure InitScreen(NeadLine:word;Background,Foreground:byte);
6: Procedure WriteScreen(x,y:word;Background,Foreground:byte;s:string);
7: Procedure WriteText(x,y:word;s:string);
8: Procedure ShowScreen;
9:
10: Implementation
11:
12: const   ScreenSize      = 80;
13: type    CharType        = record
14:           ch            : char;
15:           attr          : byte;
16:         end;
17:         BufferType      = array[1..ScreenSize] of CharType;
18:         BufferPtr       = ^BufferType;
19:         VirtualType     = record
20:           nline         : integer;
21:           BufferData    : BufferPtr;
22:         end;
23:
24: var     VirtualScreen   : VirtualType;
25:         VideoScreen     : array[1..25] of BufferType absolute $B800:0;
26:
27: Function Attribute(Background,Foreground:byte):byte;
28: begin
29:   Attribute:=(Background shl 4)+Foreground;
30: end;
31:
32: Procedure InitScreen(NeadLine:word;Background,Foreground:byte);
33: var   loop               : word;
34:       IndexPtr,DummyPtr : BufferPtr;
35: begin
36:   with VirtualScreen do begin
37:     nline:=NeadLine;
38:     GetMem(BufferData,nline*SizeOf(BufferType));
39:   end;
40:   DummyPtr:=VirtualScreen.BufferData;
41:   for loop:=1 to ScreenSize do begin
42:     DummyPtr^[loop].ch:=' ';
43:     DummyPtr^[loop].attr:=Attribute(Background,Foreground);
44:   end;
45:   IndexPtr:=DummyPtr;
46:   for loop:=2 to VirtualScreen.nline do  begin
47:     Inc(IndexPtr);
48:     IndexPtr^:=DummyPtr^;
49:   end;
50: end;
51:
52: Procedure WriteScreen(x,y:word;Background,Foreground:byte;s:string);
53: var   IndexPtr   : BufferPtr;
54:       loop       : word;
55:       attr       : byte;
```

รูปที่ 1 ยูนิต Virtual

```
begin
  if (y<=VirtualScreen.nline) and (y>0) then begin
    IndexPtr:=VirtualScreen.BufferData;
    Inc(IndexPtr,y-1);
    attr:=Attribute(Background,Foreground);
    for loop:=1 to Length(s) do begin
      if x+loop-1<=ScreenSize then begin
        IndexPtr^[x+loop-1].ch:=s[loop];
        IndexPtr^[x+loop-1].attr:=attr;
      end;
    end;
  end;
end;

Procedure WriteText(x,y:word;s:string);
var   IndexPtr   : BufferPtr;
      loop       : word;
begin
  if (y<=VirtualScreen.nline) and (y>0) then begin
    IndexPtr:=VirtualScreen.BufferData;
    Inc(IndexPtr,y-1);
    for loop:=1 to Length(s) do begin
      if x+loop-1<=ScreenSize then begin
        IndexPtr^[x+loop-1].ch:=s[loop];
      end;
    end;
  end;
end;

Procedure ScrollUp;
var   Reg        : Registers;
begin
  with Reg do begin
    ah:=$06;
    al:=1;
    bh:=0;
    ch:=0;
    cl:=0;
    dh:=24;
    dl:=79;
  end;
  Intr($10,Reg);
end;

Procedure ScrollDown;
var    Reg        : Registers;
begin
  with Reg do begin
    ah:=$07;
    al:=1;
    bh:=0;
    ch:=0;
    cl:=0;
    dh:=24;
    dl:=79;
```

รูปที่ 1 (ต่อ)

รูปที่ 1 (ต่อ)

```
111:    end;
112:    Intr($10,Reg);
113: end;
114:
115: Procedure ShowScreen;
116: var    loop,ysize,TopLine,BottomLine : word;
117:        TopPtr,BottomPtr              : SufferPtr;
118:        ch                            : char;
119: begin
120:   TopPtr:=VirtualScreen.SufferData;
121:   TopLine:=1;
122:   BottomPtr:=TopPtr;
123:   ysize:=VirtualScreen.nline;
124:   if ysize>25 then
125:     ysize:=25;
126:   if ysize>0 then begin
127:     for loop:=1 to ysize do begin
128:       Move(BottomPtr^,VideoScreen[loop][1],SizeOf(BufferType));
129:       Inc(BottomPtr);
130:     end;
131:     Dec(BottomPtr);
132:     BottomLine:=ysize;
133:     while ch<>#27 do begin
134:       ch:=ReadKey;
135:       if ch=#0 then begin
136:         ch:=ReadKey;
137:         case ch of
138:           #72 :
139:             if TopLine>1 then begin
140:               ScrollDown;
141:               Dec(TopPtr);
142:               Dec(TopLine);
143:               Dec(BottomPtr);
144:               Dec(BottomLine);
145:               Move(TopPtr^,VideoScreen[1][1],SizeOf(BufferType));
146:             end;
147:           #80 :
148:             if BottomLine<VirtualScreen.nline then begin
149:               ScrollUp;
150:               Inc(TopPtr);
151:               Inc(TopLine);
152:               Inc(BottomPtr);
153:               Inc(BottomLine);
154:               Move(BottomPtr^,VideoScreen[ysize][1],SizeOf(BufferType));
155:             end;
156:         end;
157:       end;
158:     end;
159:   end;
160: end;
161:
162: END.
```

Procedure InitScreen

สร้างจอภาพเสมือนขึ้นโดยมีจำนวนบรรทัดเท่ากับ NeadLine และใช้ Background เป็นสีพื้น Foreground เป็นสีตัวอักษร

Procedure WriteScreen

เขียนตัวอักษรบนจอภาพเสมือนที่ตำแหน่งคอลัมน์ x และบรรทัดที่ y โดยเขียนด้วย Background เป็นสีพื้น Foreground เป็นสีตัวอักษร ส่วน s เป็นข้อความที่จะเขียน ถ้ามีส่วนของข้อความอยู่นอกจอภาพ จะเขียนเฉพาะส่วนที่อยู่ในขอบเขตของจอภาพเสมือนเท่านั้น

Procedure WriteText

ทำหน้าที่เหมือนโพรซีเดอร์ WriteScreen จะแตกต่างกันในส่วนของสีพื้นและสีตัวอักษรที่ใช้ จะใช้สีเดิมของจอภาพ หมายถึงจะเขียนเฉพาะส่วนที่เป็นข้อความเท่านั้นโดยไม่เปลี่ยนแปลงแอตทริบิวต์

Procedure ShowScreen

แสดงจอเสมือนบนจอจริงโดยใช้คีย์ลูกศรขึ้นและคีย์ลูกศรลงในการเลื่อนจอภาพ และคีย์ Esc เมื่อต้องการออกจากการแสดงข้อมูลนี้

■

ตัวอย่าง

เราจะเริ่มตัวอย่างการใช้งานยูนิต Virtual ด้วยการใช้ตัวอย่างของโปรแกรม DemoScreen ในรูปที่ 2 จะเห็นว่าเราสร้างตัวแปร Message ในบรรทัดที่ 4 ให้เป็นอะเรย์ของสตริงเพื่อใช้เป็นข้อมูลตัวอักษรที่จะเขียนบนจอภาพเสมือน การใช้งานจอภาพเสมือนเริ่มจากบรรทัดที่ 48 ในรูปที่ 2 โดยเรียกใช้โพรซีเดอร์ InitScreen เพื่อสร้างจอภาพเสมือนขนาด 40 บรรทัด ใช้ตัวอักษรสีเขียวบนพื้นสีดำ หลังจากสร้างจอภาพเสมือนแล้ว โพรซีเดอร์ MakeScreen จะเขียนตัวอักษรบนจอภาพเสมือนโดยใช้โพรซีเดอร์ WriteScreen ในบรรทัดที่ 42 เขียนหัวข้อบนบรรทัดแรกคือบรรทัดที่ 1 โดยใช้ตัวอักษรสีแดงบนพื้นสีเทา และจะเขียนข้อความจากตัวแปร Message โดยใช้โพรซีเดอร์ WriteText และหลังจากนั้นก็เรียกใช้โพรซีเดอร์ ShowScreen เพื่อแสดงให้เห็นจอภาพเสมือนบนจอภาพจริง

โพรซีเดอร์ที่ให้ไว้ในยูนิต Virtual เป็นเพียงตัวอย่างเบื้องต้นเท่านั้น ในการใช้งานจริงผู้อ่านยังคงต้องพัฒนาโพรซีเดอร์ขึ้นมาเองให้เหมาะสมกับงานที่จะนำไปใช้ สวัสดีครับ!

รูปที่ 2 โปรแกรม DemoScreen

```
 1: Program DemoScreen;
 2: Uses Crt,Virtual;
 3:
 4: const    Message     :  array[1..33] of string    =
 5: ( 'Turbo Pascal is designed to meet the needs of all types of users of'
 6:   'IBM PS/2s, PCs, and compatibles. It is a structured, high-level',
 7:   'language you can use to write programs for any type or size of',
 8:   'application.',
 9:   '',
10:   'Turbo Pascal 6.0 builds on what is already the world'#39's standard'
11:   'Pascal compiler. Fully compatible with code written using earlier',
12:   'version of Turbo Pascal, This new version also includes',
13:   '',
14:   '* a brand new, state-of-the-art integrated development',
15:   '  environment (IDE), with',
16:   '  .multiple overlapping windows',
17:   '  .mouse support, menus, dialogs',
18:   '  .multi-file editor that can edit files up to 1 Mb',
19:   '  .enhanced debugging facilities',
20:   '  .complete save and restore of desktop',
21:   '* an object-oriented application framework, Turbo Vision, for use',
22:   '  in your applications (it gives you the same tools we used to',
23:   '  write the IDE)',
24:   '* a full-featured inline assembler',
25:   '  private fields and methods in object declarations',
26:   '  extended syntax directive ($X) that lets you treat functions like
27:   '  procedures (and ignore function results)',
28:   '* 286 code generation',
29:   '* address reference in types constant',
30:   '* far and near procedure directive ($G)',
31:   '* link in initialized data ($L) from object files',
32:   '* new heap manager is faster and reduce fragmentation',
33:   '  (FreeMin and FreeList have been replaceed; refer to Chapter 16 in
34:   '  the Programmer'#39's Guide for more information)',
35:   '* enhanced hypertext online help facilities, with complete cut-',
36:   '  and-paste example code for every library procedure and',
37:   '  function'   );
38:
39: Procedure MakeScreen;
40: var   loop    : word;
41: begin
42:   WriteScreen(24,1,LightGray,Red,' INTRODUCTION TO TURBO PASCAL 6.0 ')
43:   for loop:=1 to 33 do
44:     WriteText(1,2+loop,Message[loop]);
45: end;
46:
47: BEGIN
48:   InitScreen(40,Black,Green);
49:   MakeScreen;
50:   TextColor(LightGray);
51:   TextBackground(Black);
52:   ClrScr;
53:   ShowScreen;
54: END.
```